Conjuntos test en optimización entera no lineal

J. Gago M. I. Hartillo J. Puerto J.M. Ucha

Priego de Córdoba, 27-29 de Septiembre 2013

Programación entera con ayuda del álgebra

Recordaremos algunas de las ideas desarrolladas para tratar el siguiente problema **(P)**:

$$\begin{aligned} \text{mín} \quad & \mathbf{c}'\mathbf{x}, \\ \text{sujeto a} \quad & A\mathbf{x} = \mathbf{b} \\ & g_1(\mathbf{x}) \leq C_1 \\ & \vdots \\ & g_m(\mathbf{x}) \leq C_m, \\ & \\ & \mathbf{x} = (x_1, \ldots, x_n)' \in \mathbb{Z}^n_{\geq 0}, \quad m \geq 1. \end{aligned}$$

con A matriz entera. Se trata de un problema de programación entera no lineal (con función objetivo *lineal*).

Publicaciones hasta la fecha

Cartera de valores entera

An algebraic approach to integer portfolio problems. **European Journal of Operational Research 2011** CGHPU.

Fiabilidad

Exact cost minimization of a series-parallel reliable system with multiple component choices using an algebraic method., **Computers and Operations Research 2013** GHPU.

Tayur, Thomas y Natraj '1995

Nuestra inspiración inicial

Tayur, Thomas y Natraj, en *An algebraic geometry algorithm for scheduling in presence of setups and correlated demands*, [**Math. Programming (1995)**], presentaron un método para encontrar una solución exacta para un tipo de problema entero con restricción de probabilidad.

Tayur, Thomas y Natraj '1995

Nuestra inspiración inicial

Tayur, Thomas y Natraj, en *An algebraic geometry algorithm for scheduling in presence of setups and correlated demands*, [**Math. Programming (1995)**], presentaron un método para encontrar una solución exacta para un tipo de problema entero con restricción de probabilidad.

Su método se puede generalizar para cualquier **(P)** como el descrito anteriormente.
Usan una idea de Sturmfels [Gröbner bases and Convex Polytopes] para visitar, de forma ordenada, todos los puntos factibles de un problema de programación lineal entera.

Tayur, Thomas y Natraj '1995

El método se basa en:

- El cálculo de un conjunto test (test set) para un subproblema lineal $\mathbf{(P}_L\mathbf{)}$ de **(P)**.
- Un proceso de búsqueda inversa (walk-back) para alcanzar, comenzando en el óptimo de $\mathbf{(P}_L\mathbf{)}$ el óptimo de **(P)**.

Walk-back simple: conjuntos test

Definición

Un conjunto $\mathcal{G}_{>_c} \subset \mathbb{Z}^N$ es un **conjunto test** de una familia de IP con matriz A y coste $\mathbf{c}$ si

- para cada punto no óptimo α factible de IP existe $g \in \mathcal{G}_{>_c}$ tal que $\alpha - g$ es un punto factible y $\alpha >_c \alpha - g$,
- para el punto óptimo β de IP, $\beta - g$ es no factible para todo $g \in \mathcal{G}_{>_c}$.

Walk-back simple: conjuntos test

Definición

Un conjunto $\mathcal{G}_{>_c} \subset \mathbb{Z}^N$ es un **conjunto test** de una familia de IP con matriz A y coste $\mathbf{c}$ si

- para cada punto no óptimo α factible de IP existe $g \in \mathcal{G}_{>_c}$ tal que $\alpha - g$ es un punto factible y $\alpha >_c \alpha - g$,
- para el punto óptimo β de IP, $\beta - g$ es no factible para todo $g \in \mathcal{G}_{>_c}$.

Un conjunto test proporciona un método para resolver un IP dado un punto factible: en cada paso, o existe un elemento del conjunto test que mejora al actual, o no hay mejora y estamos en el óptimo.

Walk-back simple: conjuntos test

Conjuntos test algebraicos

Los conjuntos test se pueden calcular usando bases de Gröbner de ciertos *ideales tóricos* asociados a A y c.

Este paso puede ser el cuello de botella del problema, pero...

Walk-back simple: conjuntos test

Conjuntos test algebraicos

Los conjuntos test se pueden calcular usando bases de Gröbner de ciertos *ideales tóricos* asociados a A y c.

Este paso puede ser el cuello de botella del problema, pero...

Walk-back simple: conjuntos test

Conjuntos test algebraicos

Los conjuntos test se pueden calcular usando bases de Gröbner de ciertos *ideales tóricos* asociados a A y c.

Este paso puede ser el cuello de botella del problema, pero...

- Hemos encontrado muchos problemas para los que el tiempo de cálculo del conjunto test es muy razonable.

Walk-back simple: conjuntos test

Conjuntos test algebraicos

Los conjuntos test se pueden calcular usando bases de Gröbner de ciertos *ideales tóricos* asociados a A y c.

Este paso puede ser el cuello de botella del problema, pero...

- Hemos encontrado muchos problemas para los que el tiempo de cálculo del conjunto test es muy razonable.
- A veces somos capaces de dar una fórmula cerrada los conjuntos test para familias de problemas.

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $\mathbf{(P}_L\mathbf{)}$.

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $\mathbf{(P}_L\mathbf{)}$.
- Si β es factible para **(P)**, es óptimo de **(P)**.

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $\mathbf{(P}_L\mathbf{)}$.
- Si β es factible para **(P)**, es óptimo de **(P)**.
- Si β no es factible para **(P)**, usamos el esqueleto reverso $\mathcal{G}'_{>c}$.

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $\mathbf{(P}_L\mathbf{)}$.
- Si β es factible para **(P)**, es óptimo de **(P)**.
- Si β no es factible para **(P)**, usamos el esqueleto reverso $\mathcal{G}'_{>_c}$.
- Para cada γ punto obtenido mediante $\mathcal{G}'_{>_c}$:

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $\mathbf{(P}_L\mathbf{)}$.
- Si β es factible para **(P)**, es óptimo de **(P)**.
- Si β no es factible para **(P)**, usamos el esqueleto reverso $\mathcal{G}'_{>_c}$.
- Para cada γ punto obtenido mediante $\mathcal{G}'_{>_c}$:
  - Si $c(\gamma) > c_0$, podamos la rama.

Desandando el camino...

Resolviendo **(P)**

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $\mathbf{(P}_L\mathbf{)}$.
- Si β es factible para **(P)**, es óptimo de **(P)**.
- Si β no es factible para **(P)**, usamos el esqueleto reverso $\mathcal{G}'_{>c}$.
- Para cada γ punto obtenido mediante $\mathcal{G}'_{>c}$:
  - Si $c(\gamma) > c_0$, podamos la rama.
  - Si $\gamma_i < 0$ podamos la rama.

Desandando el camino...

Resolviendo (P)

- c_0 mejor coste de punto factible y^0 para **(P)**
- Calculamos β óptimo de $(\mathbf{P}_L)$.
- Si β es factible para **(P)**, es óptimo de **(P)**.
- Si β no es factible para **(P)**, usamos el esqueleto reverso $\mathcal{G}'_{>_c}$.
- Para cada γ punto obtenido mediante $\mathcal{G}'_{>_c}$:
  - Si $c(\gamma) > c_0$, podamos la rama.
  - Si $\gamma_i < 0$ podamos la rama.
  - Si γ es factible para **(P)**, y $c(\gamma) < c_0$, actualizamos c_0 e y^0.

Ventajas e incovenientes de la búsqueda

Ventajas

- La búsqueda inversa nos proporciona de forma ordenada, mediante coste creciente, todos los puntos factibles de **(P)**.

Ventajas e incovenientes de la búsqueda

Ventajas

- La búsqueda inversa nos proporciona de forma ordenada, mediante coste creciente, todos los puntos factibles de **(P)**.
- Un nuevo punto factible y^0 que mejore el coste, descarta todos los nodos pendientes de coste mayor o igual.

Ventajas e incovenientes de la búsqueda

Ventajas

- La búsqueda inversa nos proporciona de forma ordenada, mediante coste creciente, todos los puntos factibles de **(P)**.
- Un nuevo punto factible y^0 que mejore el coste, descarta todos los nodos pendientes de coste mayor o igual.

Incoveniente principal

Si los puntos factibles de **(P)** están muy lejos de β óptimo de **(LP)**, el número de nodos a procesar es enorme.

Si acortamos la región de búsqueda añadiendo restricciones lineales, el conjunto test cambia (generalmente se complica).

Problemas en la enumeración de puntos factibles

Básicamente, el método de Walk-back realiza una búsqueda en anchura moviéndose mediante el conjunto test.

Problemas en la enumeración de puntos factibles

Básicamente, el método de Walk-back realiza una búsqueda en anchura moviéndose mediante el conjunto test.

Sturmfels en "Gröbner bases and Convex polytopes"

Para evitar una lista demasiado grande de nodos *activos* durante el recorrido hacia atrás, es interesante usar la técnica de **Avis & Fukuda 1992** (por cierto, pararelizable de forma natural.)

Mejoras en la búsqueda

Procesamiento

El tamaño de la lista de nodos *activos* crece muy rápidamente, en nuestro movimiento entre los puntos factibles de $\mathbf{(P}_L\mathbf{)}$, mediante el conjunto test:

- Si los ordenamos por coste, el coste computacional es muy alto.

Mejoras en la búsqueda

Procesamiento

El tamaño de la lista de nodos *activos* crece muy rápidamente, en nuestro movimiento entre los puntos factibles de $\mathbf{(P}_L\mathbf{)}$, mediante el conjunto test:

- Si los ordenamos por coste, el coste computacional es muy alto.
- Se tarda mucho en encontrar los **mejores** puntos factible para **(P)**.

Mejoras en la búsqueda

Procesamiento

El tamaño de la lista de nodos *activos* crece muy rápidamente, en nuestro movimiento entre los puntos factibles de $\mathbf{(P_L)}$, mediante el conjunto test:

- Si los ordenamos por coste, el coste computacional es muy alto.
- Se tarda mucho en encontrar los **mejores** puntos factible para **(P)**.
- Tenemos que ordenar la lista de pendientes mediante un balance entre coste y factibilidad para **(P)**.

Notación

$$\textbf{(P)} \quad \begin{aligned} &\text{mín } c'x \\ &x \in \mathcal{A} \\ &x \in \mathcal{B} \end{aligned}$$

Notación

$$\textbf{(P)} \quad \begin{array}{l} \text{mín } c'x \\ x \in \mathcal{A} \\ x \in \mathcal{B} \end{array}$$

$$\textbf{(P)} \quad \begin{array}{l} \text{mín } c'x \\ Ax = b \\ g_1(x) \leq C_1 \\ \vdots \\ g_m(x) \leq C_m \end{array}$$

Notación

$$\textbf{(P)} \quad \begin{array}{l} \text{mín } c'x \\ x \in \mathcal{A} \\ x \in \mathcal{B} \end{array}$$

$$\textbf{(P)} \quad \begin{array}{l} \text{mín } c'x \\ Ax = b \\ g_1(x) \leq C_1 \\ \vdots \\ g_m(x) \leq C_m \end{array}$$

$$\textbf{(P}_L\textbf{)} \quad \begin{array}{l} \text{mín } c'x \\ x \in \mathcal{A} \end{array}$$

Notación

$$\textbf{(P)} \quad \begin{array}{l} \text{mín } c'x \\ x \in \mathcal{A} \\ x \in \mathcal{B} \end{array}$$

$$\textbf{(P)} \quad \begin{array}{l} \text{mín } c'x \\ Ax = b \\ g_1(x) \leq C_1 \\ \vdots \\ g_m(x) \leq C_m \end{array}$$

$$\textbf{(P}_L\textbf{)} \quad \begin{array}{l} \text{mín } c'x \\ x \in \mathcal{A} \end{array}$$

$$(P_L) \quad \begin{array}{l} \text{mín } c'x \\ Ax = b \end{array}$$

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (penalización estática)

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (penalización estática)
  - $r(x) = (\text{máx}(g_1(x) - C_1, 0), \ldots, \text{máx}(g_m(x) - C_m), 0)$

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (penalización estática)
  - $r(x) = (\text{máx}(g_1(x) - C_1, 0), \ldots, \text{máx}(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (penalización adaptativa)

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (penalización estática)
  - $r(x) = (\text{máx}(g_1(x) - C_1, 0), \ldots, \text{máx}(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (penalización adaptativa)
- $c_p(x) = c(x) \cdot (2 - D(x))$

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (penalización estática)
  - $r(x) = (\text{máx}(g_1(x) - C_1, 0), \ldots, \text{máx}(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (penalización adaptativa)
- $c_p(x) = c(x) \cdot (2 - D(x))$
  - $D(x) = (\prod d_i(x))^{1/m}$

Funciones de penalización

Función de penalización

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distancia de x a $\mathcal{B}$.
  - Si $\mathcal{B}$ viene dado por $g_i(x) \leq C_i$: $P(x) = \sum \text{máx}(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (penalización estática)
  - $r(x) = (\text{máx}(g_1(x) - C_1, 0), \ldots, \text{máx}(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (penalización adaptativa)
- $c_p(x) = c(x) \cdot (2 - D(x))$
  - $D(x) = (\prod d_i(x))^{1/m}$
  - $$d_i(x) = \begin{cases} 1 & \text{si } g_i(x) \leq C_i \\ \left| \frac{C_i}{g_i(x)} \right| & \text{si } g_i(x) > C_i \end{cases}$$

Secuenciación de trabajos

El problema tratado en Tayur, Thomas y Natraj, es un problema de secuencia de trabajos por lotes a máquinas, para minimizar costes de producción y puesta en marcha.

Secuenciación de trabajos

El problema tratado en Tayur, Thomas y Natraj, es un problema de secuencia de trabajos por lotes a máquinas, para minimizar costes de producción y puesta en marcha.
Entre las restricciones las hay de capacidad temporal de las máquinas y una restricción de probabilidad para alcanzar la demanda.

Secuenciación de trabajos

El problema tratado en Tayur, Thomas y Natraj, es un problema de secuencia de trabajos por lotes a máquinas, para minimizar costes de producción y puesta en marcha.
Entre las restricciones las hay de capacidad temporal de las máquinas y una restricción de probabilidad para alcanzar la demanda.
Esta última restricción es de probabilidad con la forma:

$$\mathrm{Prob}(\tilde{T}x \leq \mathbf{C}) \geq \gamma$$

con $\tilde{T}$ una matriz tecnológica que representa una muestra de demandas. Además hay correlación entre las variables, lo que hace más difícil el problema

Modelo

- n número de trabajos, indicados por i,

Modelo

- n número de trabajos, indicados por i,
- m número de máquinas, indicadas por j,

Modelo

- n número de trabajos, indicados por i,
- m número de máquinas, indicadas por j,
- $(D_1, \ldots, D_n)$ vector aleatorio de demandas,

Modelo

- n número de trabajos, indicados por i,
- m número de máquinas, indicadas por j,
- $(D_1, \ldots, D_n)$ vector aleatorio de demandas,
- C_j capacidad (tiempo) de cada máquina,

Modelo

- n número de trabajos, indicados por i,
- m número de máquinas, indicadas por j,
- $(D_1, \ldots, D_n)$ vector aleatorio de demandas,
- C_j capacidad (tiempo) de cada máquina,
- $(\hat{D}_1, \ldots, \hat{D}_n)$ vector de medias de la probabilidad de distribución de demanda,

Modelo

- n número de trabajos, indicados por i,
- m número de máquinas, indicadas por j,
- $(D_1, \ldots, D_n)$ vector aleatorio de demandas,
- C_j capacidad (tiempo) de cada máquina,
- $(\hat{D}_1, \ldots, \hat{D}_n)$ vector de medias de la probabilidad de distribución de demanda,
- S_{ij} tiempo de puesta en marcha para el trabajo i en máquina j,

Modelo

- n número de trabajos, indicados por i,
- m número de máquinas, indicadas por j,
- $(D_1, \ldots, D_n)$ vector aleatorio de demandas,
- C_j capacidad (tiempo) de cada máquina,
- $(\hat{D}_1, \ldots, \hat{D}_n)$ vector de medias de la probabilidad de distribución de demanda,
- S_{ij} tiempo de puesta en marcha para el trabajo i en máquina j,
- K_{ij} coste de puesta en marcha del trabajo i en máquina j,

Modelo

- M_i tamaño de lote,

Modelo

- M_i tamaño de lote,
- L'_{ij} coste de producción por unidad de i en máquina j,

Modelo

- M_i tamaño de lote,
- L'_{ij} coste de producción por unidad de i en máquina j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,

Modelo

- M_i tamaño de lote,
- L'_{ij} coste de producción por unidad de i en máquina j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,
- p_{ij} tiempo de proceso de unidad de i en máquina j,

Modelo

- M_i tamaño de lote,
- L'_{ij} coste de producción por unidad de i en máquina j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,
- p_{ij} tiempo de proceso de unidad de i en máquina j,
- γ probabilidad de alcanzar la demanda.

Modelo

- M_i tamaño de lote,
- L'_{ij} coste de producción por unidad de i en máquina j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,
- p_{ij} tiempo de proceso de unidad de i en máquina j,
- γ probabilidad de alcanzar la demanda.
- z_{ij} variable binaria trabajo i se planifica en j,

Modelo

- M_i tamaño de lote,
- L'_{ij} coste de producción por unidad de i en máquina j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,
- p_{ij} tiempo de proceso de unidad de i en máquina j,
- γ probabilidad de alcanzar la demanda.
- z_{ij} variable binaria trabajo i se planifica en j,
- y_{ij} multiplos de $1/M_i$ de demanda de producto i planificado en j.

Modelo

$$\text{(SP) minimiza } \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

$$\text{s.t.} \sum_{j=1}^{m} y_{ij} = M_i, i = 1, 2, \ldots, n, \tag{1}$$

$$M_i z_{ij} \geq y_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{2}$$

$$\sum_{i=1}^{n} p_{ij} \left(\hat{D}_i / M_i \right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m, \tag{3}$$

$$\text{Prob} \left\{ \sum_{i=1}^{n} p_{ij} \left(D_i / M_i \right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m \right\} \geq \gamma, \tag{4}$$

$$z_{ij} \in \{0, 1\}, y_{ij} \in \{0, 1, \ldots, M_i\} \tag{5}$$

Problema relajado

$$(\mathrm{LSP}) \text{ minimiza } \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

s.t.

$$\sum_{j=1}^{m} y_{ij} = M_i, i = 1, 2, \ldots, n, \tag{6}$$

$$M_i z_{ij} \geq y_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{7}$$

$$z_{ij} \in \{0, 1\}, y_{ij} \in \{0, 1, \ldots, M_i\}. \tag{8}$$

Funciones de penalización

Si $\mathbf{x}$ es el conjunto de variables y_{ij}, z_{ij}, consideraremos:

$$G_0(\mathbf{x}) = \gamma - g_0(\mathbf{x}),$$

$$g_0(\mathbf{x}) = \text{Prob}\left\{\sum_{i=1}^{n} p_{ij}\left(D_i/M_i\right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m\right\},$$

$$G_j(\mathbf{x}) = g_j(\mathbf{x}) - C_j,$$

$$g_j(\mathbf{x}) = \sum_{i=1}^{n} p_{ij}\left(\hat{D}_i/M_i\right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij}, j = 1, 2, \ldots, m.$$

Funciones de penalización

La primera función de penalización es: $T_0(\mathbf{x}) = c(\mathbf{x}) + \mu P(\mathbf{x})$

Funciones de penalización

La primera función de penalización es: $T_0(\mathbf{x}) = c(\mathbf{x}) + \mu P(\mathbf{x})$ con

$$P(\mathbf{x}) = \sum_{j=0}^{m} \text{máx}(G_j(\mathbf{x}), 0).$$

para calcular μ, consideramos

Funciones de penalización

La primera función de penalización es: $T_0(\mathbf{x}) = c(\mathbf{x}) + \mu P(\mathbf{x})$ con

$$P(\mathbf{x}) = \sum_{j=0}^{m} \text{máx}(G_j(\mathbf{x}), 0).$$

para calcular μ, consideramos

- $\mathbf{p}$ óptimo del problema relajado (LSP),

Funciones de penalización

La primera función de penalización es: $T_0(\mathbf{x}) = c(\mathbf{x}) + \mu P(\mathbf{x})$ con

$$P(\mathbf{x}) = \sum_{j=0}^{m} \text{máx}(G_j(\mathbf{x}), 0).$$

para calcular μ, consideramos

- $\mathbf{p}$ óptimo del problema relajado (LSP),
- $\rho = G_0(\mathbf{p})$, $c_0 = c(\mathbf{p})$

Funciones de penalización

La primera función de penalización es: $T_0(\mathbf{x}) = c(\mathbf{x}) + \mu P(\mathbf{x})$ con

$$P(\mathbf{x}) = \sum_{j=0}^{m} \text{máx}(G_j(\mathbf{x}), 0).$$

para calcular μ, consideramos

- $\mathbf{p}$ óptimo del problema relajado (LSP),
- $\rho = G_0(\mathbf{p})$, $c_0 = c(\mathbf{p})$
- $c_1 = \sum_i \sum_j (K_{ij} + M_i L_{ij})$

Funciones de penalización

La primera función de penalización es: $T_0(\mathbf{x}) = c(\mathbf{x}) + \mu P(\mathbf{x})$ con

$$P(\mathbf{x}) = \sum_{j=0}^{m} \text{máx}(G_j(\mathbf{x}), 0).$$

para calcular μ, consideramos

- $\mathbf{p}$ óptimo del problema relajado (LSP),
- $\rho = G_0(\mathbf{p})$, $c_0 = c(\mathbf{p})$
- $c_1 = \sum_i \sum_j (K_{ij} + M_i L_{ij})$

$$\mu_0 = \frac{c_1 - c_0}{\rho}, \alpha = \lfloor \log(\frac{c_0}{\mu_0}) \rfloor, \text{ y } \mu = \alpha\mu_0.$$

Funciones de penalización

La segunda función de penalización es:

$$T_1(\mathbf{x}) = c(\mathbf{x})(2 - D(\mathbf{x}))$$

Funciones de penalización

La segunda función de penalización es:

$$T_1(\mathbf{x}) = c(\mathbf{x})(2 - D(\mathbf{x}))$$

Con $D(\mathbf{x}) = \left(\prod_{j=0}^{m} d_j(\mathbf{x})\right)^{1/(m+1)}$ y

Funciones de penalización

La segunda función de penalización es:

$$T_1(\mathbf{x}) = c(\mathbf{x})(2 - D(\mathbf{x}))$$

Con $D(\mathbf{x}) = \left(\prod_{j=0}^{m} d_j(\mathbf{x})\right)^{1/(m+1)}$ y

$$d_0(\mathbf{x}) = \begin{cases} 1 & \text{si } g_0(\mathbf{x}) \geq \gamma, \\ \frac{g_0(\mathbf{x})}{\gamma} & \text{si } g_0(\mathbf{x}) < \gamma. \end{cases}$$

$$d_j(\mathbf{x}) = \begin{cases} 1 & \text{si} g_j(\mathbf{x}) \leq C_j, \\ \frac{C_j}{g_j(\mathbf{x})} & \text{si } g_j(\mathbf{x}) \geq C_j. \end{cases} \quad j = 1, 2, \ldots, m,$$

Funciones de penalización

La segunda función de penalización es:

$$T_1(\mathbf{x}) = c(\mathbf{x})(2 - D(\mathbf{x}))$$

Con $D(\mathbf{x}) = \left(\prod_{j=0}^{m} d_j(\mathbf{x})\right)^{1/(m+1)}$ y

$$d_0(\mathbf{x}) = \begin{cases} 1 & \text{si } g_0(\mathbf{x}) \geq \gamma, \\ \frac{g_0(\mathbf{x})}{\gamma} & \text{si } g_0(\mathbf{x}) < \gamma. \end{cases}$$

$$d_j(\mathbf{x}) = \begin{cases} 1 & \text{si} g_j(\mathbf{x}) \leq C_j, \\ \frac{C_j}{g_j(\mathbf{x})} & \text{si } g_j(\mathbf{x}) \geq C_j. \end{cases} \quad j = 1, 2, \ldots, m,$$

Resultados computacionales

Cuadro : 4 máquinas, 7 trabajos, $M = 2$, modelo original

	Walk-back				Penalty T_0			
	Total		Óptimo		Total		Óptimo	
γ	T	Nodos	T	Nodos	T	Nodos	T	Nodos
0,888	148,7	13708	91,2	12409	54,3	8861	0,8	596
0,900	149,3	13709	92,1	12410	54,8	8865	0,8	600
0,932	237,3	18560	29,2	7132	196,3	18498	1,1	777
0,956		Max		NNP		Max	6194,5	83857
0,960		Max		NNP		Max	14397,0	116797
0,980		Max		NNP		Max		NNP

Resultados computacionales

Cuadro : 4 máquinas, 7 trabajos, $M = 2$, modelo original

	Walk-back				Penalty T_1			
	Total		Óptimo		Total		Óptimo	
γ	T	Nodos	T	Nodos	T	Nodos	T	Nodos
0,888	148,7	13708	91,2	12409	54,1	8972	1,1	733
0,900	149,3	13709	92,1	12410	53,9	8931	1,1	737
0,932	237,3	18560	29,2	7132	192,8	18516	1,3	898
0,956		Max		NNP		Max	239,7	18184
0,960		Max		NNP		Max	5614,5	78284
0,980		Max		NNP		Max		NNP

Mejora del modelo

El modelo original admite una desigualdad lineal adicional

$$z_{ij} \leq y_{ij}, i = 1, \ldots, n, j = 1, \ldots, m. \tag{9}$$

Mejora del modelo

El modelo original admite una desigualdad lineal adicional

$$z_{ij} \leq y_{ij}, i = 1, \ldots, n, j = 1, \ldots, m. \tag{9}$$

Añadir esta restricción disminuye el espacio de búsqueda

Mejora del modelo

El modelo original admite una desigualdad lineal adicional

$$z_{ij} \leq y_{ij}, i = 1, \ldots, n, j = 1, \ldots, m. \tag{9}$$

Añadir esta restricción disminuye el espacio de búsqueda
El cálculo de la base de Gröbner del modelo lineal mejorado sigue siendo muy rápido (menos de 1 s)

Resultados computacionales

Cuadro : 4 máquinas, 7 trabajos, $M = 2$, modelo mejorado

	Walk-back				Penalty T_0			
	Total		Óptimo		Total		Óptimo	
γ	T	Nodos	T	Nodos	T	Nodos	T	Nodos
0,888	3,4	934	1,3	873	2,8	643	0,2	100
0,900	3,4	934	1,3	873	2,7	643	0,2	100
0,932	5,4	1299	1,4	893	4,1	913	0,1	63
0,956	176,9	15220	130,2	14670	64,8	8511	2,1	1145
0,960	534,9	28575	429,3	27799	263,7	19072	62,5	7973
0,980		Max		NNP	6355,7	98294	11,7	3360

Resultados computacionales

Cuadro : 4 máquinas, 7 trabajos, $M = 2$, modelo mejorado

	Walk-back				Penalty T_1			
	Total		Óptimo		Total		Óptimo	
γ	T	Nodos	T	Nodos	T	Nodos	T	Nodos
0,888	3,4	934	1,3	873	2,7	641	0,1	97
0,900	3,4	934	1,3	873	2,7	641	0,2	97
0,932	5,4	1299	1,4	893	4,1	913	0,1	63
0,956	176,9	15220	130,2	14670	60,9	8064	1,2	727
0,960	534,9	28575	429,3	27799	194,1	16172	22,4	4497
0,980		Max		NNP	6316,1	98920	87,5	9905

Comparativa con otros resolutores

Otros resolutores

- El resolutor COUENNE devuelve el mensaje "System error" y para la ejecución sin devolver ningún punto.

Comparativa con otros resolutores

Otros resolutores

- El resolutor COUENNE devuelve el mensaje "System error" y para la ejecución sin devolver ningún punto.
- BARON devuelve un punto que no es factible para la restricción de probabilidad, y no proporciona ningún mensaje sobre ello.

Comparativa con otros resolutores

Otros resolutores

- El resolutor COUENNE devuelve el mensaje "System error" y para la ejecución sin devolver ningún punto.
- BARON devuelve un punto que no es factible para la restricción de probabilidad, y no proporciona ningún mensaje sobre ello.
- BONMIN no trata problemas con región factible no convexa, pero podría devolver (como heurístico) un punto factible. Sin embargo, el proceso acaba con un punto no factible para la restricción de probabilidad.